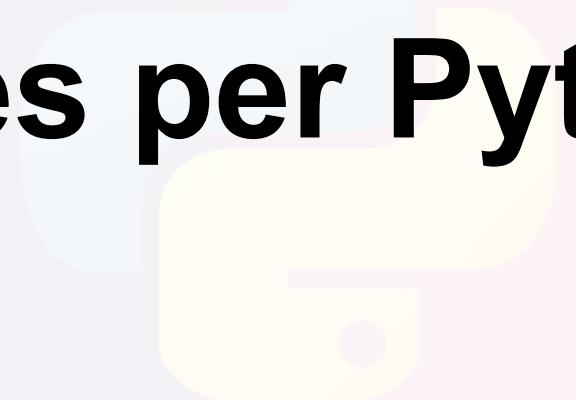


Introducció Eines per Python



Tools



Tools



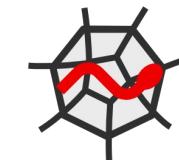
1. Jupyter Notebook



3. Libs



SciPy



2. IDE



matplotlib



seaborn



Guido van Rossum

A Short History Of Python

Conceived Python



Late 80s

Dec 1989

Feb 1991

Jan 1994

Release Python 2.0



Oct 2000

Release Python 3.7



Jun 2018

Started implementation



Release Python 1.0



Release Python 3.0



1. Jupyter Notebook

- Què és?
- Com s'instal·la?
- Com s'utilitza?
- Com funciona internament?
- Gràfiques i Widgets
- Exemples



Què és?

Jupyter Notebook es una aplicació web que et permet crear i compartir documents que continguin:

- codi executable (per exemple Python, entre d'altres)
- visualitzacions
- text explicatiu (escrit en sintaxis markdown)

El nom del projecte Jupyter es una referència als tres principals llenguatges de programació suportats: Julia, Python y R,

Història



IPython

Inici del projecte a títol personal. Es tracta d'una terminal interactiva que afegeix funcionalitats extres al mode interactiu de Python, com autocompletat, resaltat de colors..



2001



Fernando Pérez
(Medellín, Colombia)



2014

Jupyter Notebook

Es crea com a projecte derivat de IPython. Es tracta d'un llenguatge agnòstic que soporta varis entorns d'execució (coneguts com a nuclis)



JupyterHub

S'exten el notebook a servidors de multiusuari. Permet administrar i gestionar múltiples execucions en el núvol.



2016



2018

JupyterLab

S'actualiza a una més potent i flexible interfície d'usuari dels clàssics Jupyter Notebooks (notebook, terminal, editor de text, explorador de fitxers, texts enriquit..)

Per a què el puc fer servir?

Jupyter Notebook és molt útil en els següents casos:

- aprendre i provar Python
- processament i transformació de dades
- compartir codi i execucions
- aprenentatge computacional

Com s'instal·la?

El primer pas per iniciar és visitar la web del projecte a: <http://www.jupyter.org>:

Bàsicament hi trobareu dues opcions

- Prova online en el navegador
- Instalar-ho localment



Prova Online

La primera opció de “provar-ho en el navegador” us permet accedir a una versió de Jupyter Notebook allotjada al núvol. Podreu veure quina aparença té i fer alguna prova ràpida sense haver de instal·lar-ho en el vostre PC.

Thanks to Google Cloud, OVH, GESIS Notebooks and the Turing Institute for supporting us! 🎉

 binder



Starting repository: ipython/ipython-in-depth/master
Read our advice for speeding up your Binder launch.

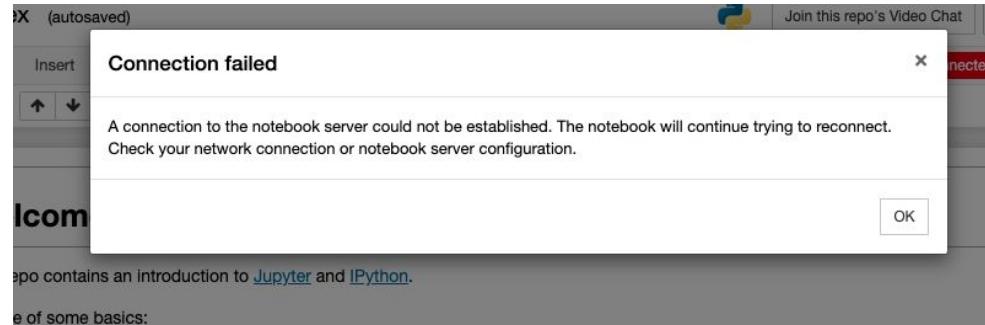
Build logs show

Prova Online

Ho recomanem tan sols per saber com és i donar-li una ullada.

Per què?

Si passeu gaire estona sense activitat, el servidor tallarà la vostra connexió i haureu percut el que s'hagi fet.



Instal·lació en local a la pròpia màquina

La segona opció és “Instalar el Notebook” on hi trobareu instruccions detallades de la instal·lació. Hi ha dues maneres principalment:

- Instalar Jupyter Notebook a través de pip (gestor de paquets de Python)

```
$ pip install jupyter
```

- Instalar a través de Anaconda (una distribució de codi obert de Python)

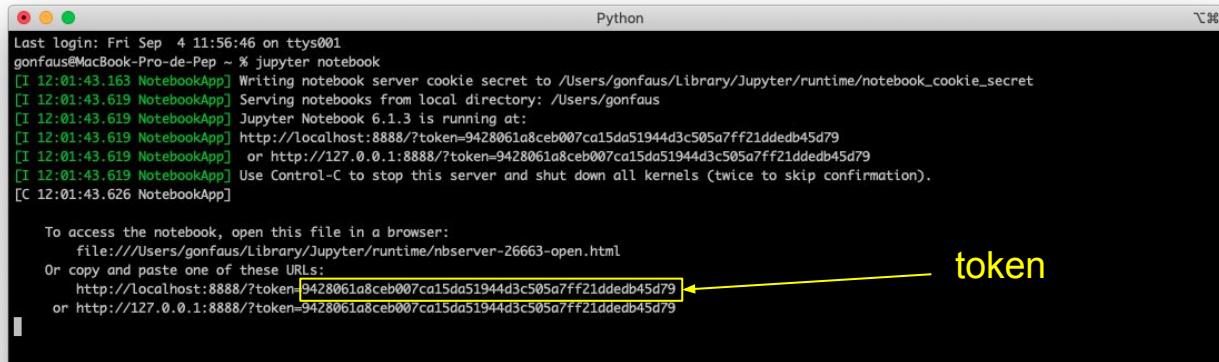
```
$ conda install jupyter
```

Iniciar Jupyter Notebook

Un cop instal·lat amb qualsevol de les 2 maneres, invocarem el servidor amb la següent comanda:

```
$ jupyter notebook
```

La consola mostrarà logs similars a això i s'obrirà una finestra del navegador per defecte al servei establert.



```
Last login: Fri Sep 4 11:56:46 on ttys001
gonfaus@MacBook-Pro-de-Pep ~ % jupyter notebook
[I 12:01:43.163 NotebookApp] Writing notebook server cookie secret to /Users/gonfaus/Library/Jupyter/runtime/notebook_cookie_secret
[I 12:01:43.619 NotebookApp] Serving notebooks from local directory: /Users/gonfaus
[I 12:01:43.619 NotebookApp] Jupyter Notebook 6.1.3 is running at:
[I 12:01:43.619 NotebookApp] http://localhost:8888/?token=9428061a8ceb007ca15da51944d3c505a7ff21ddedb45d79
[I 12:01:43.619 NotebookApp] or http://127.0.0.1:8888/?token=9428061a8ceb007ca15da51944d3c505a7ff21ddedb45d79
[I 12:01:43.619 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[12:01:43.626 NotebookApp]

To access the notebook, open this file in a browser:
file:///Users/gonfaus/Library/Jupyter/runtime/nbserver-26663-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=9428061a8ceb007ca15da51944d3c505a7ff21ddedb45d79
or http://127.0.0.1:8888/?token=9428061a8ceb007ca15da51944d3c505a7ff21ddedb45d79
```

token

Aturar Jupyter Notebook

Per aturar el servei de Jupyter Notebook i aturar tots els kernels executant-se es pot realitzar de dues formes:

- Des de la web amb el botó Quit

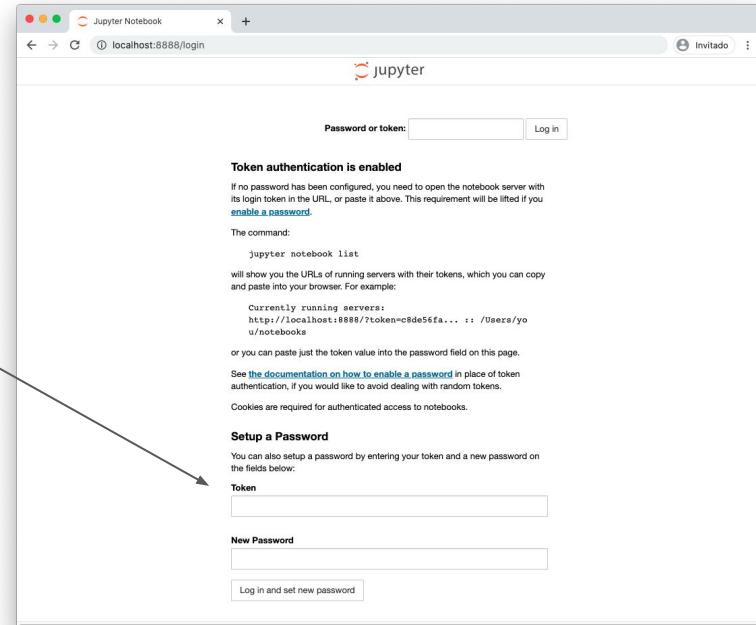
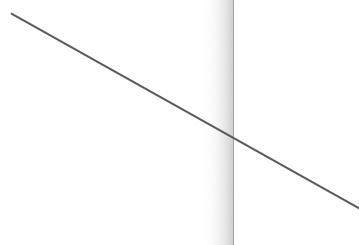


- Des de el terminal on s'ha iniciat, amb la comanda Control-C (dos cops per evitar la confirmació)

Ús del token

Si volem tornar a accedir a la url, per temes de privacitat i seguretat, us demanarà el ús del token per accedir-hi. El podeu trobar al terminal.

Per tal de no requerir el token cada cop, també podeu configurar un password de la vostra elecció



The page contains the following text:

Token authentication is enabled
If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

The command:
`jupyter notebook list`
will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:
`Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/yo
u/notebooks`
or you can paste just the token value into the password field on this page.

See [the documentation on how to enable a password](#) in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to notebooks.

Setup a Password
You can also setup a password by entering your token and a new password on the fields below.

Token
New Password
Log in and set new password

Iniciar Jupyter Notebook

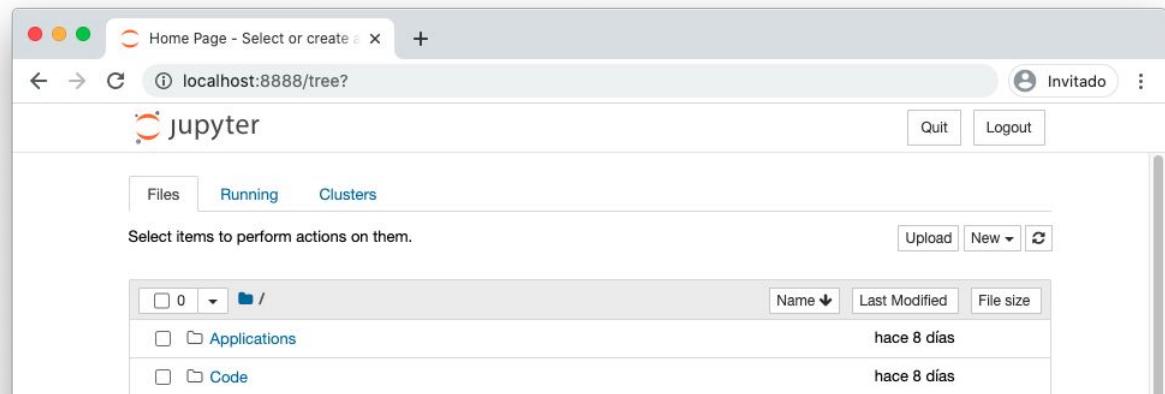
Per defecte, el navegador s'obrirà a la següent url:

http://localhost:8888

o bé

http://127.0.0.1:8888

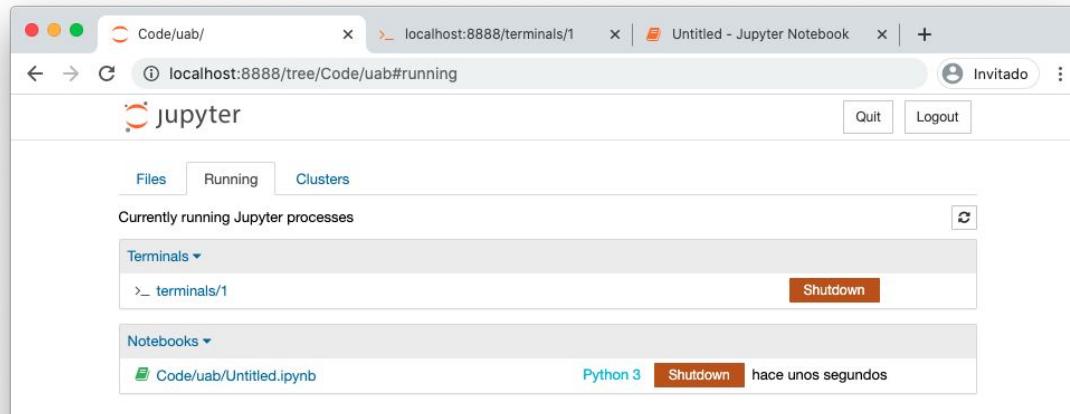
hi trobareu els continguts
de la vostra carpeta
de usuari \$home



Running Tab

Podreu veure quins kernels teniu executant actualment a la pestanya de Running. Des de aquí també les podeu aturar.

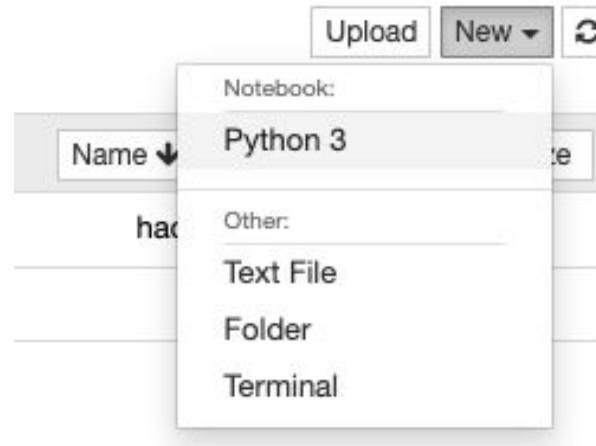
Podrieu trobar-vos que si obriu molts notebooks que carreguen moltes dades us quedeu sense memòria fàcilment.



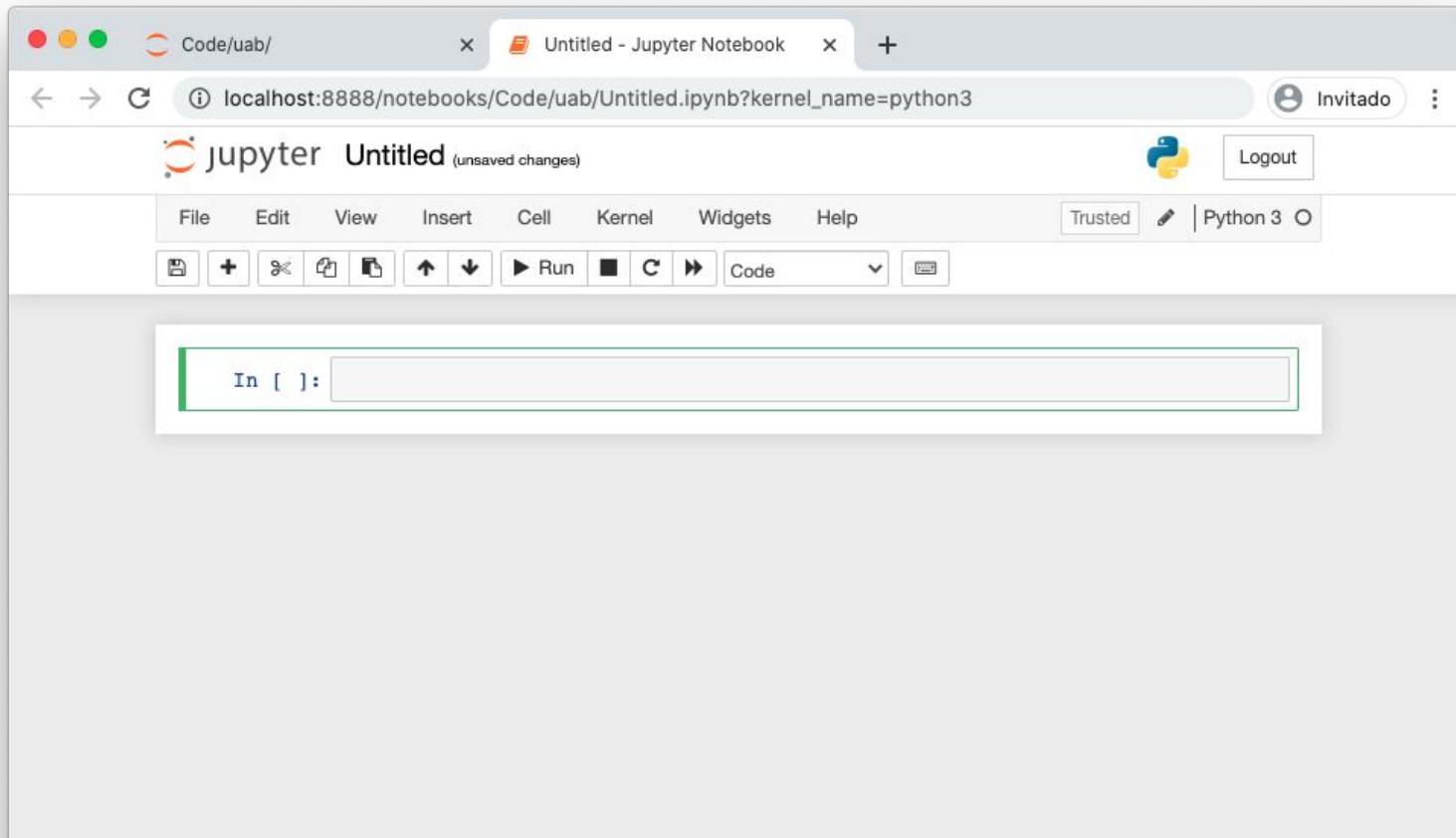
El meu primer notebook

Quan creeu un notebook en python 3:

- es crea un fitxer .ipynb
- s'executa un kernel en segon pla
- s'obre una nova pestanya amb el fitxer en blanc

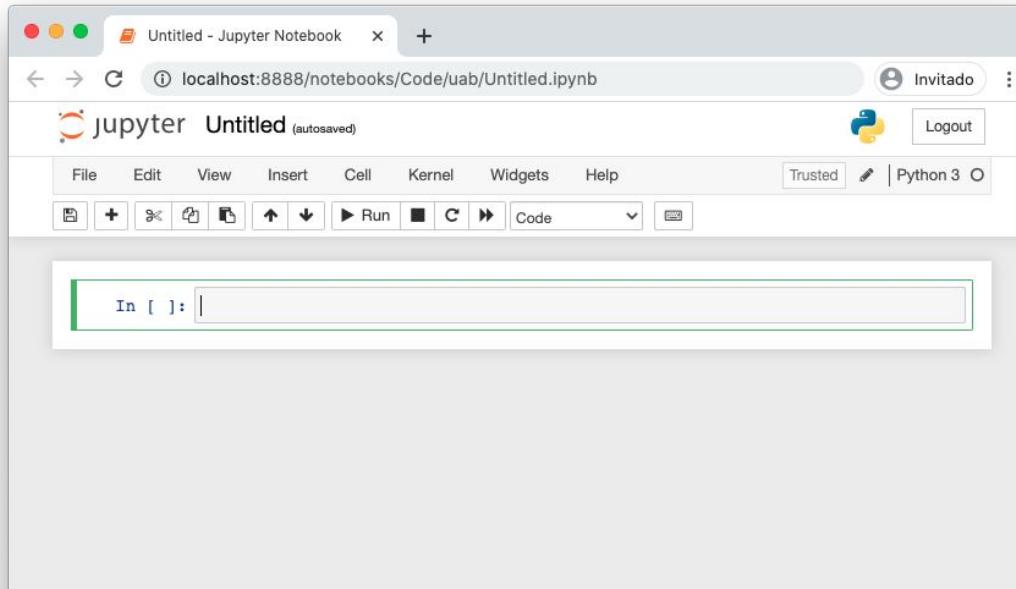


El meu primer notebook



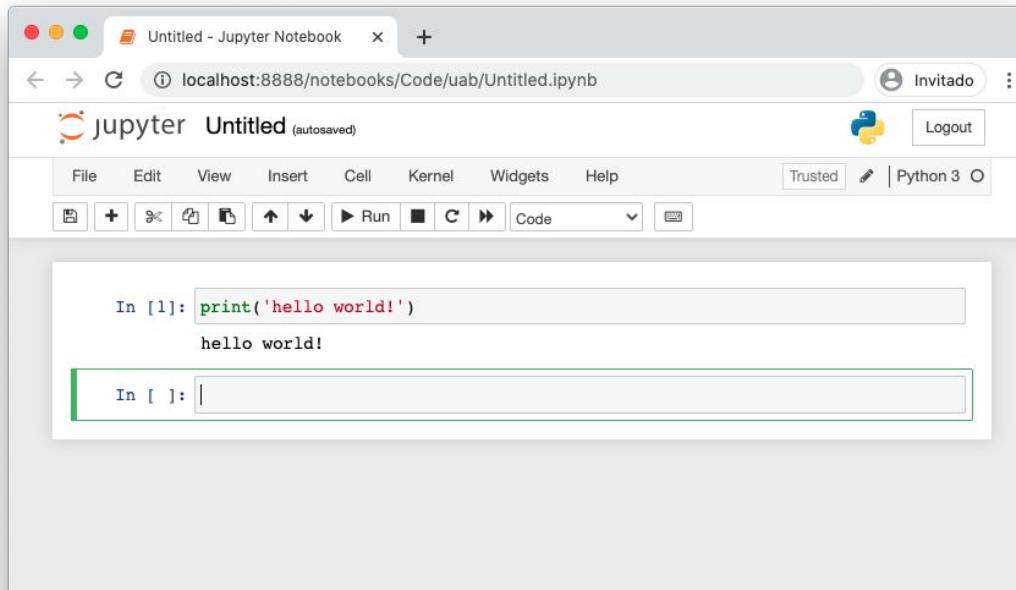
Com s'utilitza

El notebook es construeix a partir de cel·les. Inicialment, es crea amb una cel·la buida del tipus codi. Ja podem començar a escriure codi Python.



Com s'utilitza

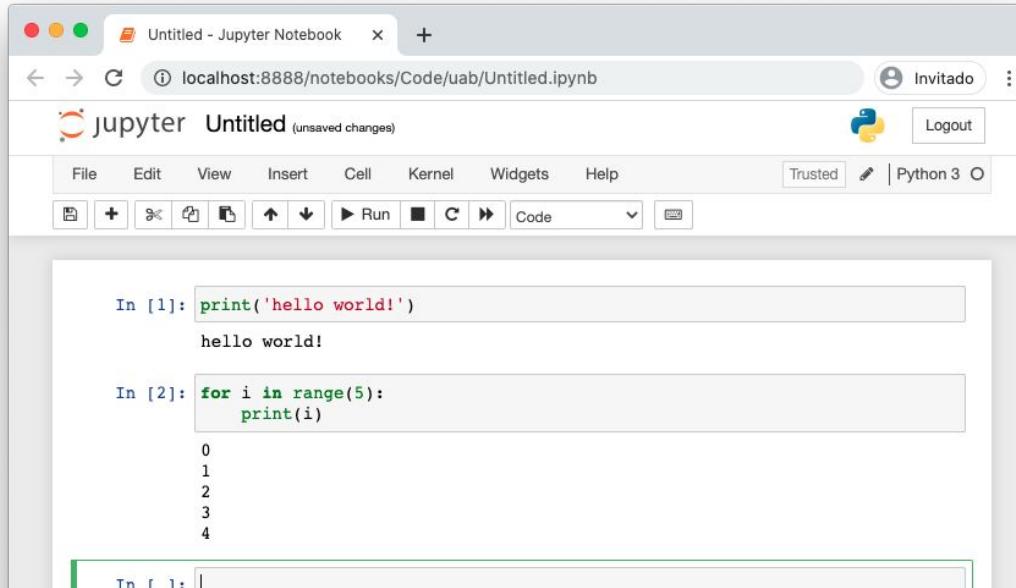
Per executar una cel·la, hem clicar el botó RUN o bé a través de la combinació de tecles Shift + Return. El resultat de la execució apareix just sota la cel·la.



Com s'utilitza

La següent cel·la es crea just a continuació i ja es pot continuar amb més codi.

Un altre exemple:



The screenshot shows a Jupyter Notebook window titled "Untitled - Jupyter Notebook". The browser tab is "localhost:8888/notebooks/Code/uab/Untitled.ipynb". The notebook interface includes a toolbar with file operations, a Python 3 kernel selector, and a toolbar below the menu bar with icons for cell creation, run, and code selection. Two code cells are visible:

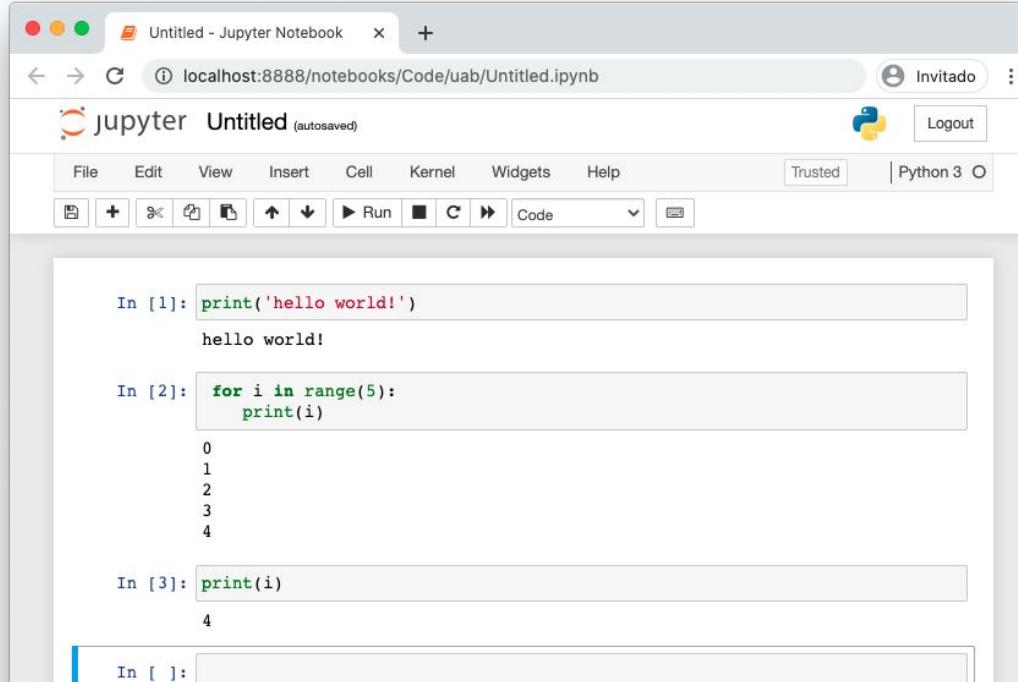
```
In [1]: print('hello world!')  
hello world!
```

```
In [2]: for i in range(5):  
    print(i)  
0  
1  
2  
3  
4
```

The output of the second cell is a list of integers from 0 to 4.

Com s'utilitza

Podem editar qualsevol de les cel·les que hem creat, tot i que s'haurà de tornar a executar per veure'n els canvis. Per exemple:



The screenshot shows a Jupyter Notebook window with the title "Untitled - Jupyter Notebook". The browser address bar displays "localhost:8888/notebooks/Code/uab/Untitled.ipynb". The notebook interface includes a toolbar with file operations like New, Open, Save, and Run, along with kernel selection (Python 3) and a Trusted button.

The notebook content consists of three code cells:

- In [1]:** `print('hello world!')`
Output: hello world!
- In [2]:** `for i in range(5):
 print(i)`
Output:
0
1
2
3
4
- In [3]:** `print(i)`
Output:
4

A fourth cell, In [], is partially visible at the bottom.

Com s'utilitza

Afegim !! a la primera cel·la i tornem a executar-la. Podem veure com el resultat s'ha modificat, però també el número al costat del codi que s'ha executat

```
In [4]: print('hello world!!!')
hello world!!!

In [2]: for i in range(5):
    print(i)

    0
    1
    2
    3
    4

In [3]: print(i)
4

In [ ]:
```

Com s'utilitza

Canviem el rang del for i tornem a executar la cel·la. S'ha actualitzat correctament, igual que abans, però el print(i) de fora el for no...

```
In [4]: print('hello world!!!')
hello world!!!

In [6]: for i in range(6):
    print(i)
0
1
2
3
4
5

In [3]: print(i)
4
```

Com s'utilitza

Encara que es modifiquin i s'executin les cel·les anteriors, fins que no s'executi l'actual no es modificarà. Per això és important entendre el nombre al costat del codi

```
In [4]: print('hello world!!!')
hello world!!!

In [6]: for i in range(6):
    print(i)
0
1
2
3
4
5

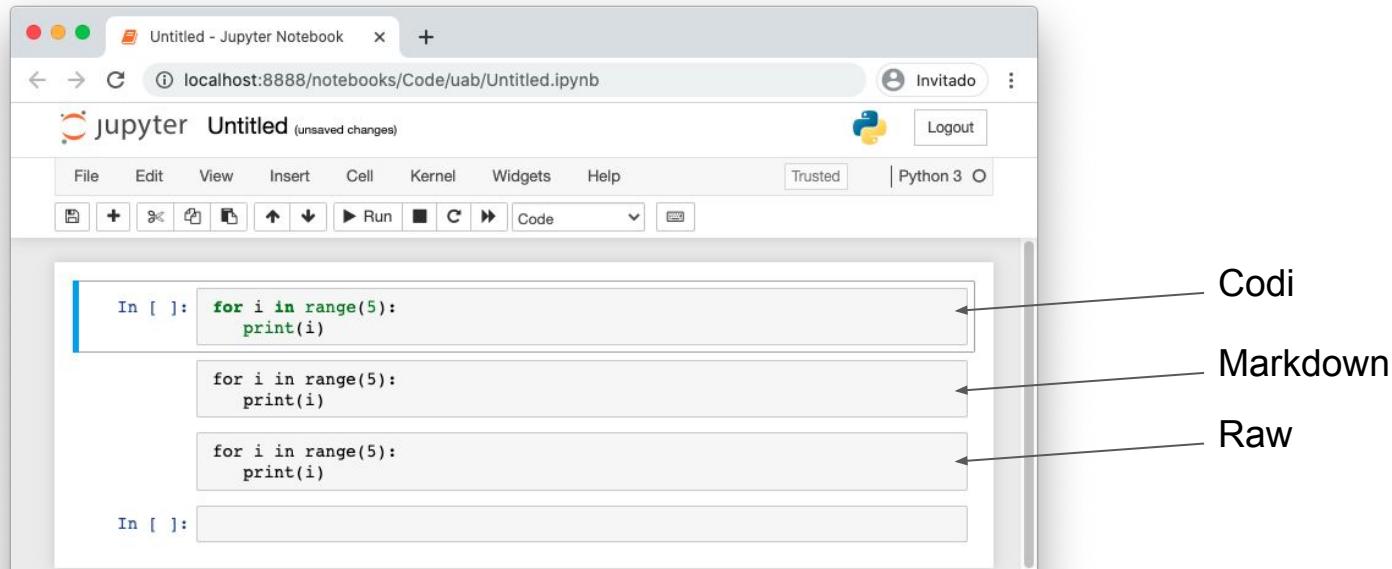
In [7]: print(i)
5
```

execution_count

Mentre s'està executant el
execution_count es converteix en *

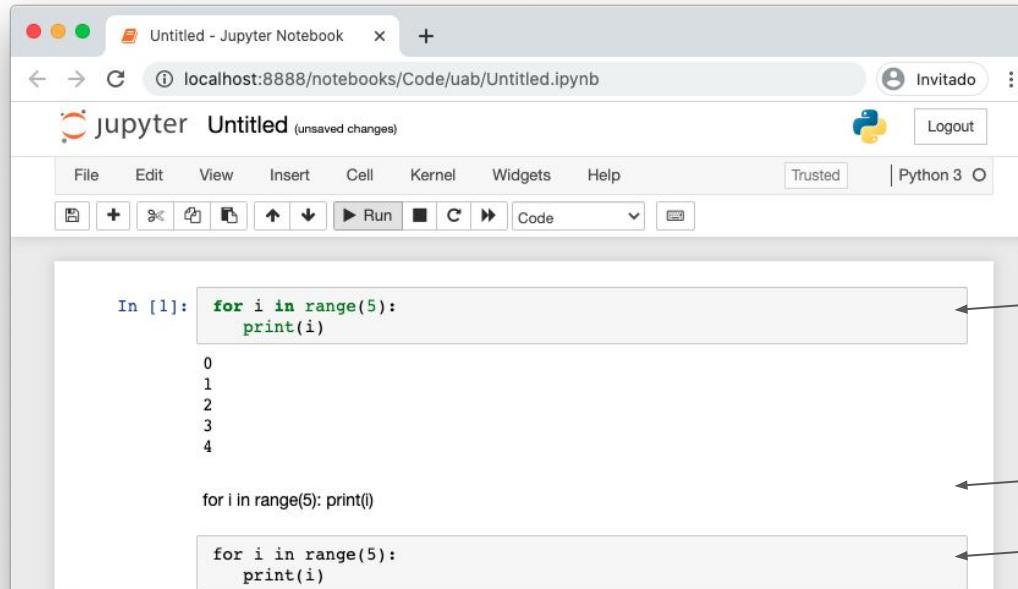
Com s'utilitza

Existeixen diferents tipus de cel·la que es poden fer servir. Per exemple, provem la mateixa entrada pels diferents tipus: el tipus Codi (el que hem fet servir fins ara), el tipus Markdown (text enriquit) o bé el tipus Raw NBConvert.



Com s'utilitza

Un cop executada la cel·la, veiem una sortida força diferent per cada un dels tipus. El tipus codi executa el text i retorna la sortida que hauríem obtingut. El tipus Markdown transforma el text al seu format i el tipus Raw ho inclou sense modificacions.



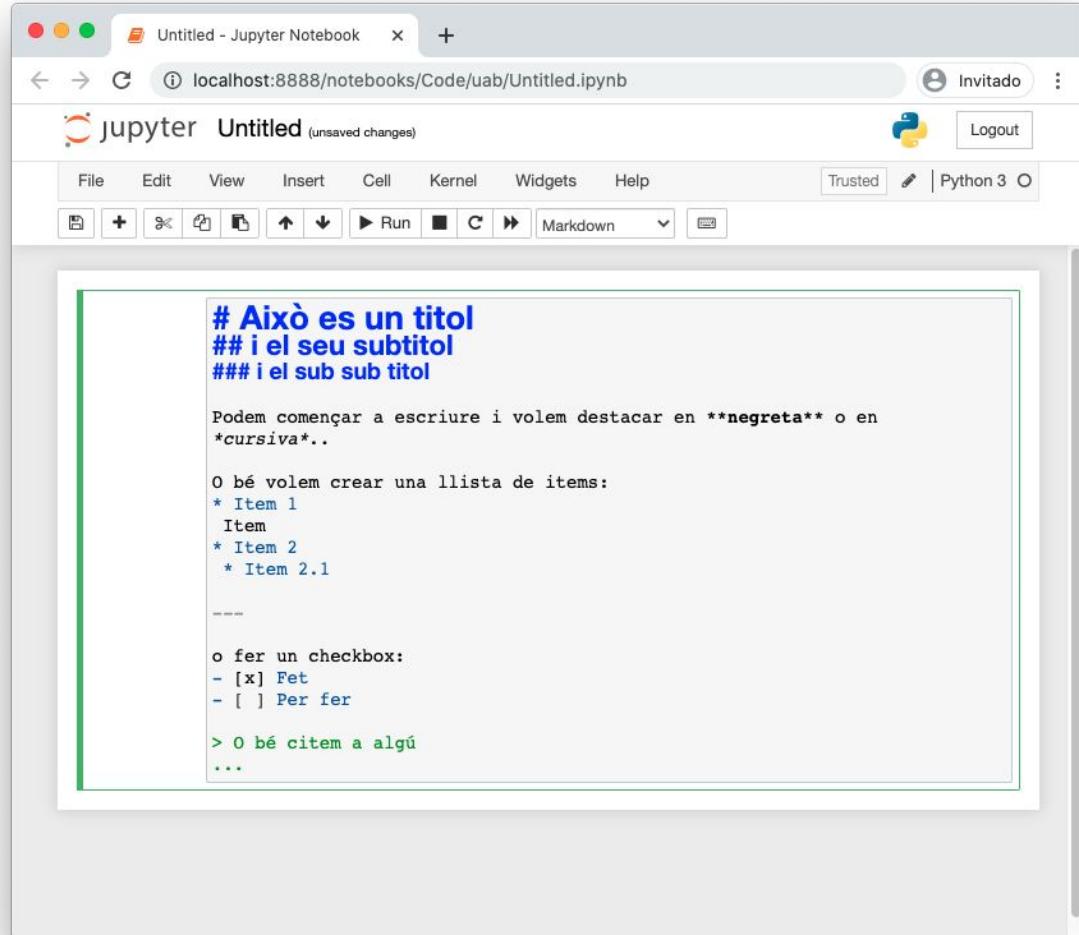
The screenshot shows a Jupyter Notebook interface with three code cells. The first cell, labeled 'In [1]:', contains Python code: `for i in range(5): print(i)`. The second cell contains the output of this code: the numbers 0 through 4. The third cell contains the same code again: `for i in range(5): print(i)`. Three arrows point from labels to specific parts of the interface:

- An arrow points to the code in the first cell with the label "Codi".
- An arrow points to the output of the first cell with the label "Markdown".
- An arrow points to the code in the third cell with the label "Raw".

Markdown

El tipus de cel·la Markdown és molt útil per formatejar text i fer-lo més lleigible. Es pot incloure latex i html també.

Per exemple, podem veure com de simple es escriure amb format:



The screenshot shows a Jupyter Notebook interface with the title "Untitled - Jupyter Notebook". The URL in the address bar is "localhost:8888/notebooks/Code/uab/Untitled.ipynb". The notebook contains the following Markdown content:

```
# Això es un titol
## i el seu subtítol
### i el sub sub titol

Podem començar a escriure i volem destacar en **negreta** o en *cursiva*.

O bé volem crear una llista de items:
* Item 1
  Item
* Item 2
  * Item 2.1

-----
o fer un checkbox:
- [x] Fet
- [ ] Per fer

> O bé citem a algú
...
```

Markdown

És un llenguatge de marques lleuger, originalment creat per John Gruber i Aaron Swartz[†] el 2004 que permet "escriure utilitzant un format de text pla fàcil d'escriure i de llegir i després convertir-ho en un XHTML o HTML estructuralment vàlid"

The screenshot shows a Jupyter Notebook window titled "Untitled - Jupyter Notebook" at the URL "localhost:8888/notebooks/Code/uab/Untitled.ipynb". The notebook contains the following Markdown content:

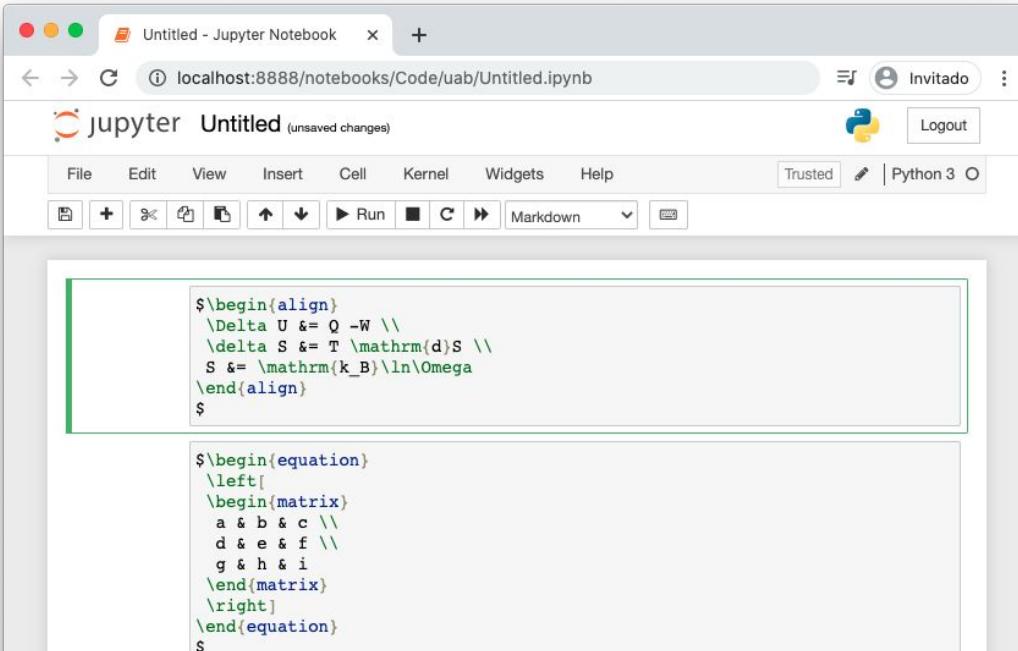
```
Això es un titol  
i el seu subtítol  
i el sub sub titol  
Podem començar a escriure i volem destacar en negreta o en cursiva..  
O bé volem crear una llista de items:  
• Item 1 Item  
• Item 2  
    • Item 2.1  
  
o fer un checkbox:  
•  Fet  
•  Per fer  
  
    O bé citem a algú ...
```

# <u>Header 1</u>	Header 1
Header 1 =====	
## <u>Header 2</u>	Header 2
Header 2 -----	
### <u>Header 3</u>	Header 3
#### <u>Header 4</u>	Header 4
##### <u>Header 5</u>	Header 5
italics	<i>italics</i>
<u>italics</u>	
literal asterisks	*literal asterisks*
<u>bold</u>	bold
<u>bold</u>	
~~strikethrough~~	strikethrough
1. <u>First item</u> 2. <u>Second item</u> <u>Subitem</u>	1. First item 2. Second item A. Subitem
* <u>Item 1</u> <u>Indent</u> - <u>Item 2</u> <u>Item 3</u>	<ul style="list-style-type: none"> • Item 1 Indent • Item 2 <ul style="list-style-type: none"> ■ Item 3
- [x] Done - [] To do	<ul style="list-style-type: none"> • <input checked="" type="checkbox"/> Done • <input type="checkbox"/> To do
A Line <u>u</u> Brake	A Line Brake
---	---
* * *	*

 [Go to anchor] (#anchor)	Go to anchor
# <u>Top Header</u> [Go to header] (#Top-Header)	
https://sqlbak.com	
[Link] (https://sqlbak.com "optional title")	Link
Click [here][id] [id]:https://sqlbak.com	
> blockquote text	blockquote text
```python print('hello'); ```	<pre>print('hello');</pre>
`inline_code();`	
Left Center Right   :---- :---- :----:  1 A C   2 B D	Left Center Right
	1 A C
	2 B D
![alt text](logo.png "Title")	
![] [id] [id]:logo.png "Title"	
\$\$\sqrt{k}\$\$ Inline: \$\sqrt{k}\$	$\sqrt{k}$
[!Img Alt Text](http://img.youtube.com/vi/ aZCXoW707nc/0.jpg)] (https://youtu.be/aZCXoW707nc "Video Title")	

# Markdown en latex

Podem escriure en latex si l'encapsulem entre “\$” en una cel·la de markdown

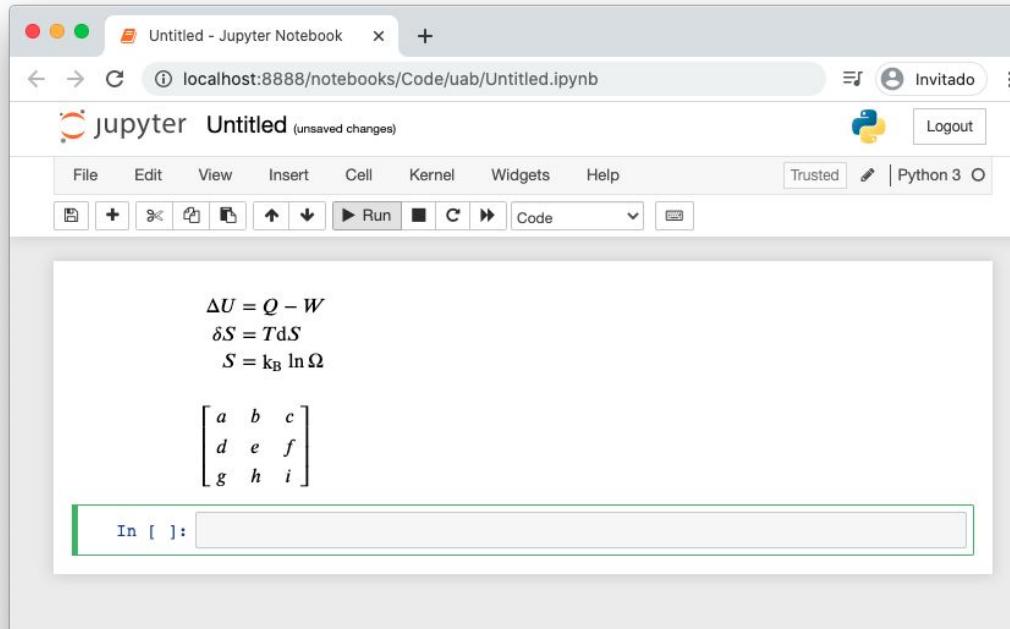


```
$\begin{aligned} \Delta U &= Q - W \\ \Delta S &= T \ln(\frac{V_f}{V_i}) \\ S &= k_B \ln(\Omega) \end{aligned}$
```

```
$\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$
```

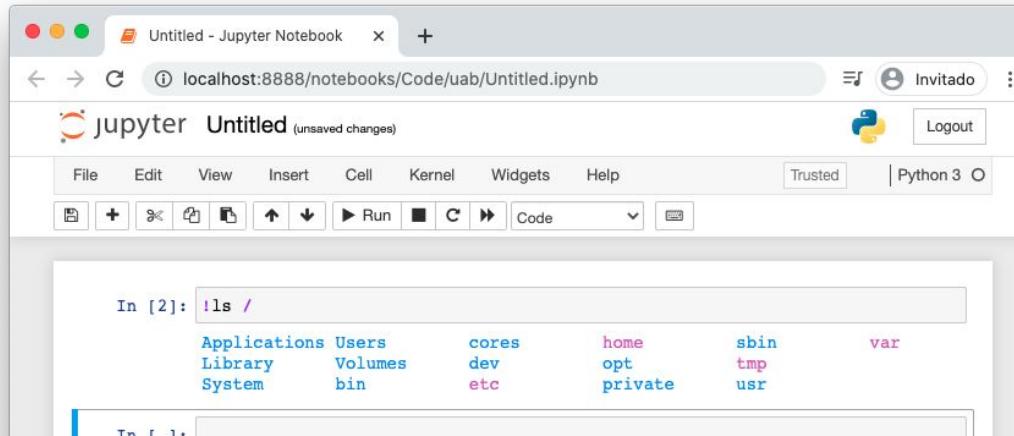
# Markdown en latex

Podem escriure en latex si l'encapsulem entre “\$” en una cel·la de markdown



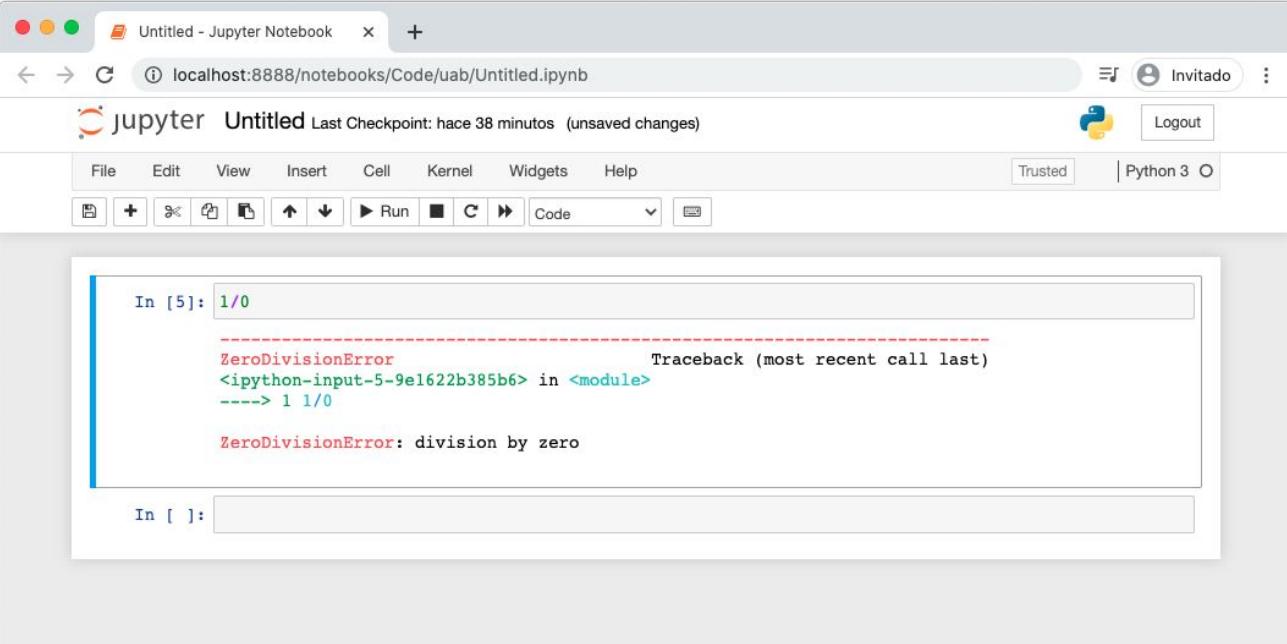
# Accés a la terminal

A través de afegir “!” en una cel·la de codi, podem accedir a comandes del terminal



# Com es gestionen els errors?

Els errors i excepcions es capturen i es mostren directament a la mateixa sortida



The screenshot shows a Jupyter Notebook window titled "Untitled - Jupyter Notebook". The URL in the address bar is "localhost:8888/notebooks/Code/uab/Untitled.ipynb". The notebook interface includes a toolbar with File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Trusted Python 3 kernel indicator. Below the toolbar is a toolbar with various icons for file operations like Open, Save, and Run. The main workspace displays a code cell labeled "In [5]:" containing the expression "1/0". A red dashed box highlights the resulting error output:

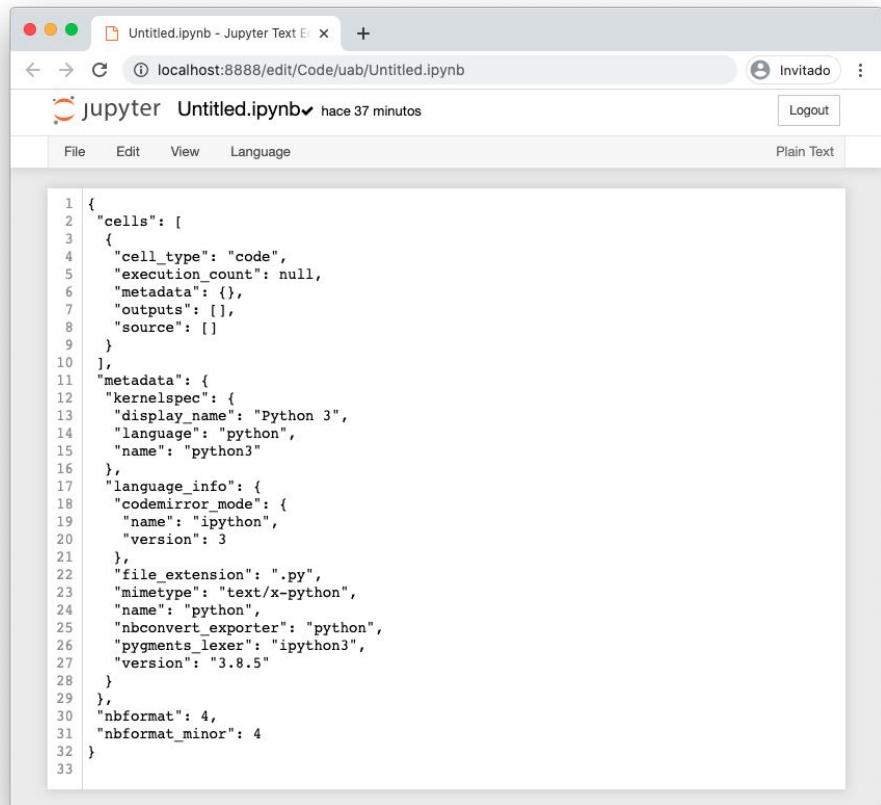
```
ZeroDivisionError Traceback (most recent call last)
<ipython-input-5-9e1622b385b6> in <module>
----> 1 1/0

ZeroDivisionError: division by zero
```

Below the error cell is another empty cell labeled "In [ ]:".

# Què conté un fitxer .ipynb?

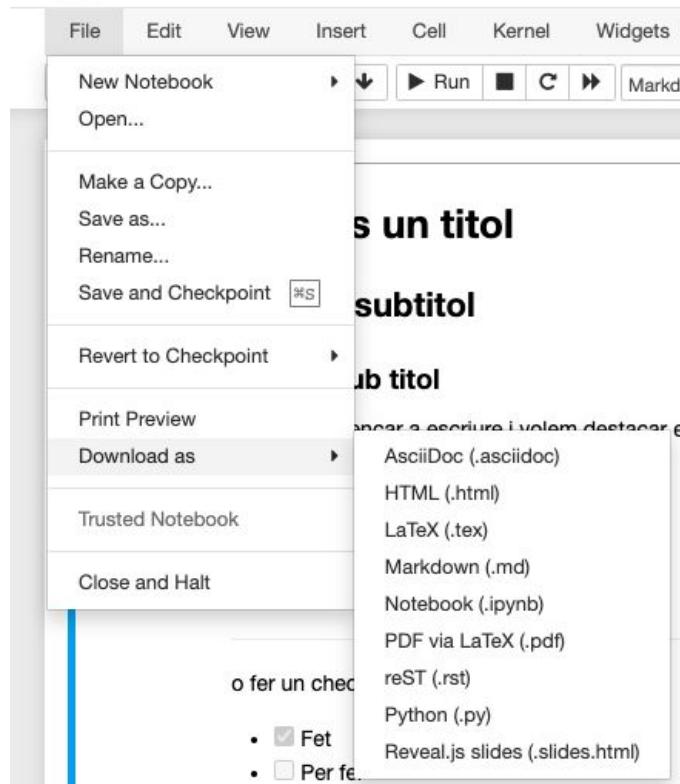
Els fitxers de Jupyter notebook són fitxers en format JSON. Cada cel·la (que n'hi ha de diferents tipus) conté el seu propi contingut (codi font, text en markdown, gràfiques, fórmules, contingut multimedia, o metadata.. ) i també els paràmetres necessaris per a la seva execució. Les cel·les estan ordenades, i poden tenir contingut d'entrada o de sortida (per emmagatzemar els resultats de la execució)



The screenshot shows a Jupyter Notebook interface with the title "Untitled.ipynb - Jupyter Text Editor". The browser address bar shows "localhost:8888/edit/Code/uab/Untitled.ipynb". The top right corner has a "Logout" button. The main area displays the JSON code of the notebook:

```
1 { "cells": [2 { "cell_type": "code", 3 "execution_count": null, 4 "metadata": {}, 5 "outputs": [], 6 "source": [] 7 } 8], 9 "metadata": { 10 "kernelspec": { 11 "display_name": "Python 3", 12 "language": "python", 13 "name": "python3" 14 }, 15 "language_info": { 16 "codemirror_mode": { 17 "name": "ipython", 18 "version": 3 19 }, 20 "file_extension": ".py", 21 "mimetype": "text/x-python", 22 "name": "python", 23 "nbconvert_exporter": "python", 24 "pygments_lexer": "ipython3", 25 "version": "3.8.5" 26 }, 27 "nbformat": 4, 28 "nbformat_minor": 4 29 } 30 }
```

# Exportació

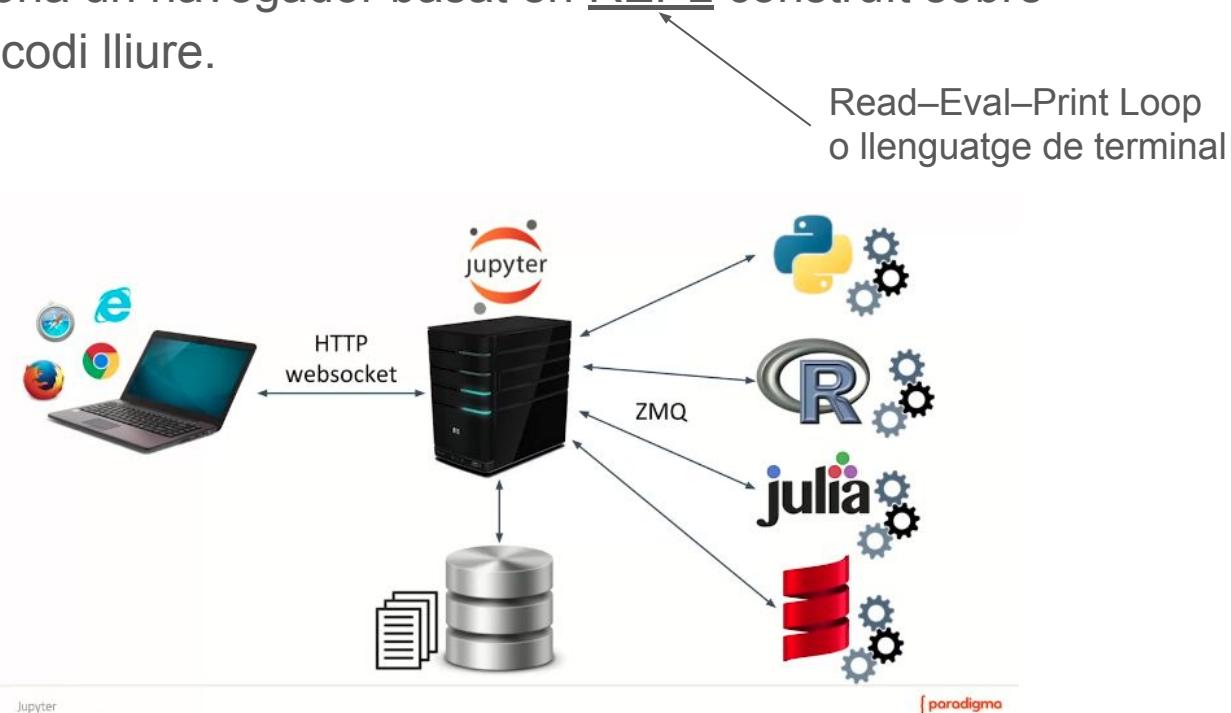


Es pot exportar fàcilment a varis formats oberts, com ara HTML, latex, PDF, Markdown i a python a través de la web i la llibreria NBCONVERT

# Com funciona internament?

Jupyter Notebook proporciona un navegador basat en REPL construït sobre varies llibreries popular de codi lliure.

- IPython
- ØMQ (zeroMQ)
- Tornado
- jQuery
- Bootstrap
- MathJax



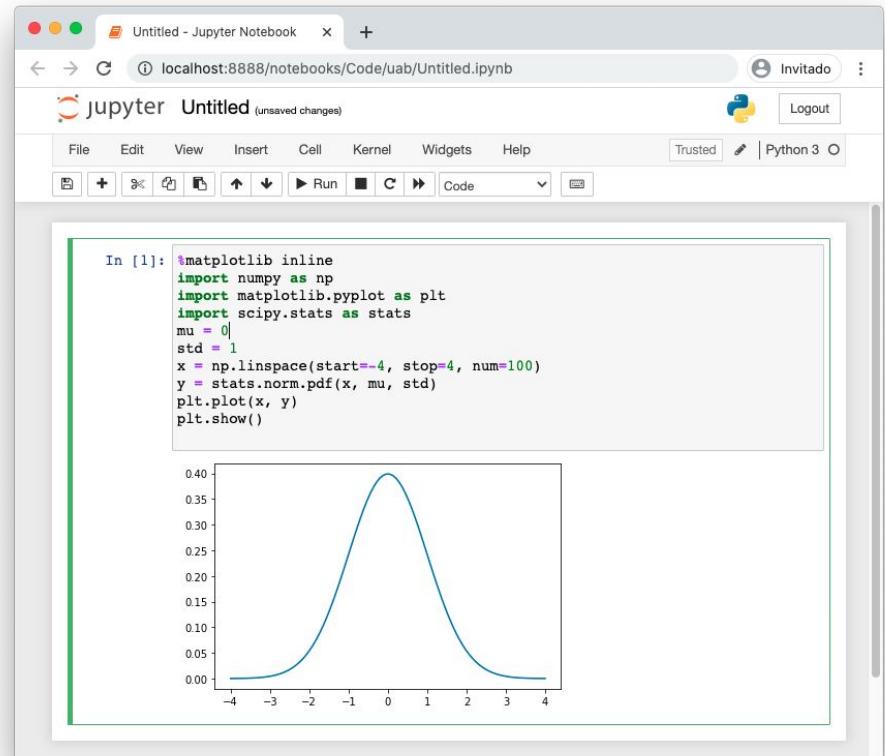
# Gràfiques

El més comú és fer servir la llibreria **matplotlib** com a visualitzador, però no és l'única (per exemple `seaborn`).

Es recomana definir la macro a l'inici del document per a que les gràfiques es mostrin dins el document

`%matplotlib inline` ← més estable,  
gràfiques estàtiques

`%matplotlib notebook` ← més dinàmic, però  
amb algun bug..



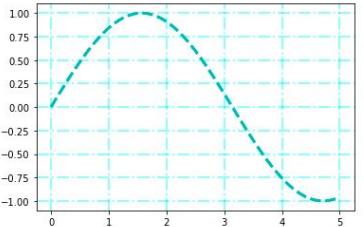
# Gràfiques

Untitled - Jupyter Notebook

jupyter Untitled (autosaved)

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 5, 100)
plt.plot(x, np.sin(x), 'c--', linewidth=3)
plt.grid(b=True, color='aqua', alpha=0.3, linestyle='-.', linewidth=2)
plt.show()
```



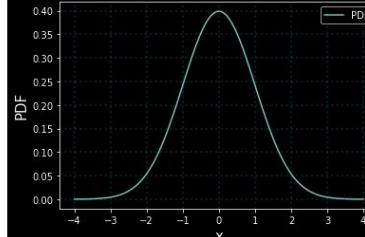
In [ ]:

Untitled - Jupyter Notebook

jupyter Untitled (unsaved changes)

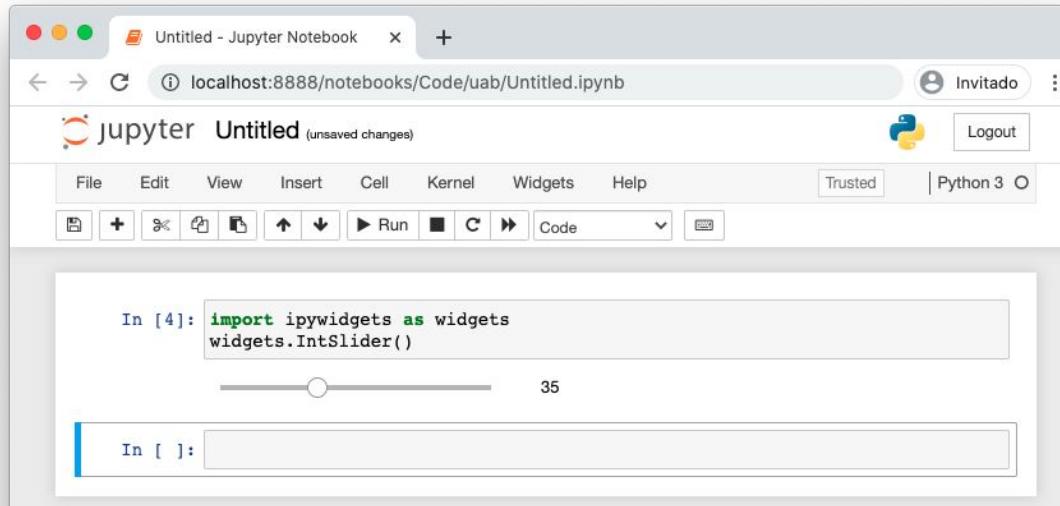
In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
plt.style.use('dark_background')
mu = 0
std = 1
x = np.linspace(start=-4, stop=4, num=100)
y = stats.norm.pdf(x, mu, std)
plt.plot(x, y, label='PDF')
plt.xlabel('x', fontsize=15)
plt.ylabel('PDF', fontsize=15)
plt.grid(b=True, color='DarkTurquoise', alpha=0.2, linestyle=':', linewidth=2)
plt.legend()
plt.show()
```



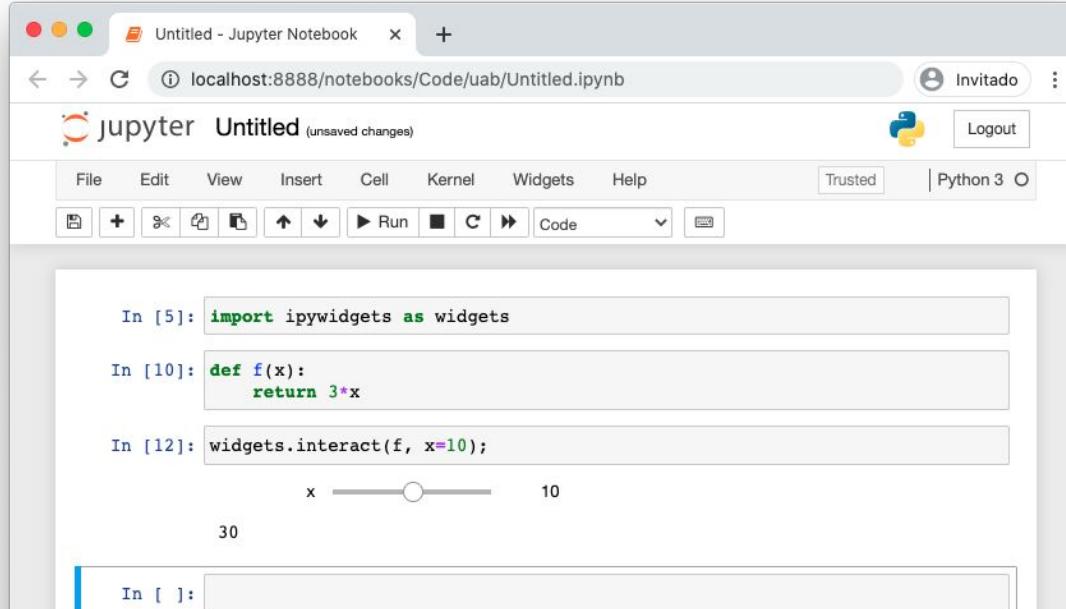
# Widgets

Són objectes de python basats en events i que tenen una representació en el navegador. S'utilitzen per paràmetres, i al utilitzarlos es llança un event que recalcula la cel·la.



# Widgets - Ús bàsic de Interact

Interact et permet delegar totes les funcions de events i quan es modifica algun valor del widget, es torna a cridar a la funció  $f(x)$  amb el nou valor



The screenshot shows a Jupyter Notebook interface with the following content:

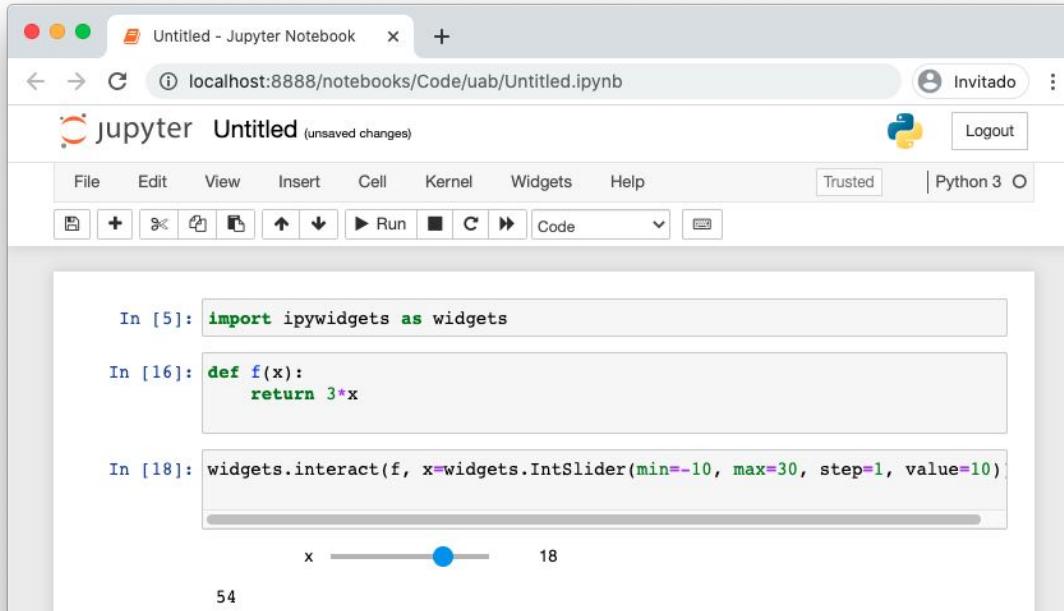
```
In [5]: import ipywidgets as widgets
In [10]: def f(x):
 return 3*x
In [12]: widgets.interact(f, x=10);
```

Below the code cells, there is a horizontal slider with the label "x" on the left and the value "10" on the right. The slider has a circular handle in the middle.

At the bottom of the notebook, there is a footer bar with the text "In [ ]:".

# Widgets - Ús bàsic de Interact

Es poden especificar els widgets directament en les variables de la funció



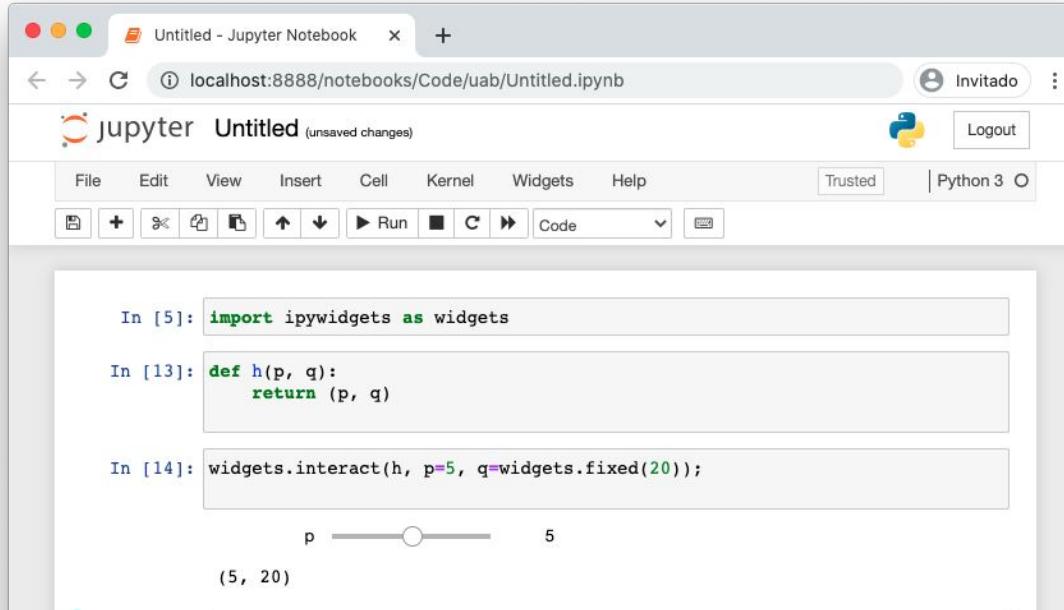
The screenshot shows a Jupyter Notebook interface with the following content:

```
In [5]: import ipywidgets as widgets
In [16]: def f(x):
 return 3*x
In [18]: widgets.interact(f, x=widgets.IntSlider(min=-10, max=30, step=1, value=10))
```

Below the code cells, there is an interactive slider with the variable 'x' labeled above it. The slider has a blue circular handle set at the value 18, which corresponds to the value of 'x' defined in the third cell.

# Widgets - Ús bàsic de Interact

Es poden passar multiples parameters a la funció, però tots han de ser utilitzats.  
Es pot utilitzar la funció fixed.



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [5]: import ipywidgets as widgets
In [13]: def h(p, q):
 return (p, q)
In [14]: widgets.interact(h, p=5, q=widgets.fixed(20));
```

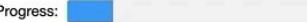
Below the code cells, there is an output cell showing a slider and its value:

p —————— 5

(5, 20)

# Widgets

Hi ha molts tipus, similars als formularis HTML

<b>IntSlider</b> Slider:  8	<b>RadioButtons</b> Options: <input checked="" type="radio"/> option 1 <input type="radio"/> option 2 <input type="radio"/> option 3	<b>Checkbox</b> <input type="checkbox"/> Check me
<b>BoundedIntText</b> Bounded Int: <input type="text" value="4"/>	<b>SelectMultiple</b> Options: <input type="checkbox"/> option 1 <input checked="" type="checkbox"/> option 2 <input type="checkbox"/> option 3	<b>Button</b> 
<b>Text</b> String: <input type="text" value="Hello World!"/>	<b>DatePicker</b> Pick a Date <input type="text" value="dd/mm/yyyy"/>	<b>Output</b>
<b>Textarea</b> String: <input type="text" value="Hello World!"/>	<b>IntProgress</b> Progress: 	<b>Play (Animation) widget</b>
<b>Dropdown</b> Number: <input type="text" value="1"/>		<b>Date picker</b>

## Widget List

- ⊕ Numeric widgets
- ⊕ Boolean widgets
- ⊕ Selection widgets
- ⊕ String widgets
- Image
- Button
- Output
- Play (Animation) widget
- Date picker
- Color picker
- File Upload
- Controller
- ⊕ Container/Layout widgets

# Ús a la indústria

- Binder
- Kaggle Kernels
- Google Colaboratory (Colab)
- Amazon SageMaker notebooks
- Azure Notebook de Microsoft
- Cocalc
- Datalore
- Github (rendering only)

# Links interessants

pàgina oficial: <https://jupyter.org/try>

documentació oficial: <https://jupyter-notebook.readthedocs.io/en/stable/index.html>

visor de jupyter notebooks online: <https://nbviewer.jupyter.org/>

pandas + widgets selectors: <https://towardsdatascience.com/bring-your-jupyter-notebook-to-life-with-interactive-widgets-bc12e03f0916>

un altre tipus de visualitzador: <https://hub.gke.mybinder.org/user/bqplot-bqplot-ea1d1yvr/notebooks/examples/Index.ipynb>

Llista interessant de notebooks: <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>

Extensions interessants: <https://towardsdatascience.com/bringing-the-best-out-of-jupyter-notebooks-for-data-science-f0871519ca29>

Slides desde notebook: <https://github.com/damianavila/RISE>

## 2. IDE

- Què és?
- Recomenacions



# Què són els IDEs?

IDE significa Integrated Development Environment

A part de editar text ens permet (entre d'altres):

- executar codi
- debugar codi
- terminals remotes
- ressaltat de sintaxis
- control de versions
- veure mètriques de rendiment
- autocompletats
- ...

Spyder

The screenshot shows the Spyder Python IDE interface. The main area is a code editor displaying a script named `temp.py`. The script contains several sections of code, including data generation, calculations, and plotting. The code uses NumPy, SciPy, and Matplotlib libraries. The variable explorer on the right shows various objects defined in the script, such as arrays, DataFrames, and plots. Below the code editor is a Python console window showing a 3D surface plot generated from the data.

```

6
7 import pylab
8 from numpy import cos, linspace, pi, sin, random
9 from scipy.interpolate import splprep, splev
10
11 # %% Generate data for analysis
12
13 # Make ascending spiral in 3-space.
14 t = linspace(0, 1.75 * 2 * pi, 100)
15
16 x = -sin(t)
17 y = cos(t)
18 z = t
19
20 # Add noise
21 x += random.normal(scale=0.1, size=x.shape)
22 y += random.normal(scale=0.1, size=y.shape)
23 z += random.normal(scale=0.1, size=z.shape)
24
25
26 # %% Perform calculations
27
28 # Spline parameters
29 smoothness = 3.0 # Smoothness parameter
30 k_param = 2 # Spline order
31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
32
33 # Find the knot points
34 knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)
35
36 # Evaluate spline, including interpolated points
37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
38
39
40 # %% Plot results
41
42 # TODO: Rewrite to avoid code smell
43 pylab.subplot(2, 2, 1)
44 data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
45 fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
46 pylab.legend()
47 pylab.xlabel('x')
48 pylab.ylabel('y')
49
50 pylab.subplot(2, 2, 2)
51 data, = pylab.plot(x, z, 'bo-', label='Data with X-Z Cross Section')
52 fit, = pylab.plot(xnew, znew, 'r-', label='Fit with X-Z Cross Section')
53 pylab.legend()
54 pylab.xlabel('x')

```

Name	Type	Size	Value
array_int8	int8	(2, 3)	Min: -9 Max: 6
array_uint32	uint32	(2, 2, 3)	Min: 1 Max: 7
bars	container.BarContainer	20	BarContainer object of matplotlib.container.
df	DataFrame	(3, 2)	Column names: bools, ints
filename	str	1	C:\ProgramData\Anaconda\lib/site-packs...
list_test	list	2	[Dataframe, Numpy array]
nrows	int	1	344
r	float64	1	7.611082589334296
radii	float64	(28,)	Min: 0.4080366385535687 Max: 9.056840974942591
region	tuple	2	(slice, slice)
rgb	float64	(45, 45, 4)	Min: 0.0 Max: 1.0
series	Series	(1,)	Series object of pandas.core.series.mod...
text_nano	NoneType	1	NoneType object of builtins module

Python console:

```

...: ls = LightSource(270, 45)
...: # To use a custom hillshading mode, override the built-in shading
...: # in the 'rgb' colors of the shaded surface calculated from "shade".
...: # rgb = ls.shade(z, cmap=cmaps.gist_earth, vert_exag=0.1, blend_mode='soft')
...: # surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, facecolors=rgb,
...: # linewidth=0, antialiased=False, shade=False)
...: ...
...: plt.show()

```

In [12]:

The screenshot shows the PyCharm IDE interface with the following components:

- Project View:** Shows the file structure of the project "djtp_first_steps".
- Code Editor:** Displays the file `tests.py` containing Django test cases for a poll application.
- Database Browser:** Shows the "Django default" database connection with tables like `auth_group`, `auth_permission`, and `django_admin_log`.
- Debugger:** Shows the current stack trace and variable values for the test case `test_index_view_with_a_future_question`.
- Bottom Status Bar:** Shows build status ("Tests Failed: 4 passed, 3 failed (4 minutes ago)"), run configuration ("Run"), and other toolbars.

# PyCharm

File Edit Selection View Go Debug Terminal Help example.py - Demo - Visual Studio Code

example.py x

```
7 #%% [markdown]
8 # Creating dataframe
9
10 Run Cell | Run Above | Debug cell
11 #%%
12 df = pd.DataFrame(data=np.random.randn(2000,
13 index=pd.date_range('2001-01-01', periods=periods, other="non existent"),
14 columns=['A', 'B']))
15
16 def gm(df, const):
17 v = (((df.A + df.B) + 1).cumprod()) - 1 * const
18 return v.iloc[-1]
19
Run Cell | Run Above
#%% [markdown]
```

periods  
Undefined variable: 'periods' Python(undefined-variable)  
Peek Problem No quick fixes available

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL Filter. E.g.: text, **/*.ts, !**/node... ⚙️ ↻ ⌂ ⌄ ×

example.py 1  
⚠ Undefined variable: 'periods' Python(undefined-variable) [12, 47]

Python 3.7.3 64-bit 0 ▲ 1 Ln 16, Col 28 Spaces: 4 UTF-8 CRLF Python 😊 📡

# Visual Studio Code

### 3. Libs

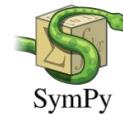
- Llibreries
- Exemples d'ús



**SciPy**



IP[y]:  
IPython



matplotlib



# Què és SciPy?

SciPy (pronunciat "Sigh Pie") és un ecosistema de programari de codi obert basat en Python per a matemàtiques, ciències i enginyeria.

Conté en el core:



IP[y]:  
IPython



Però també existeixen add-ons (3rd-party), anomenats SciKits (SciPy Toolkits).



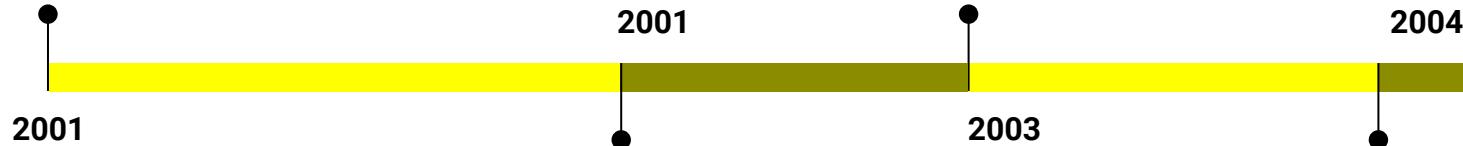
sponsored by

**NUMFOCUS**  
OPEN CODE • BETTER SCIENCE

# Història SciPy



John Hunter



Travis Oliphant

Also founder of Anaconda, NumFOCUS, creator of numpy

IP[y]:  
IPython

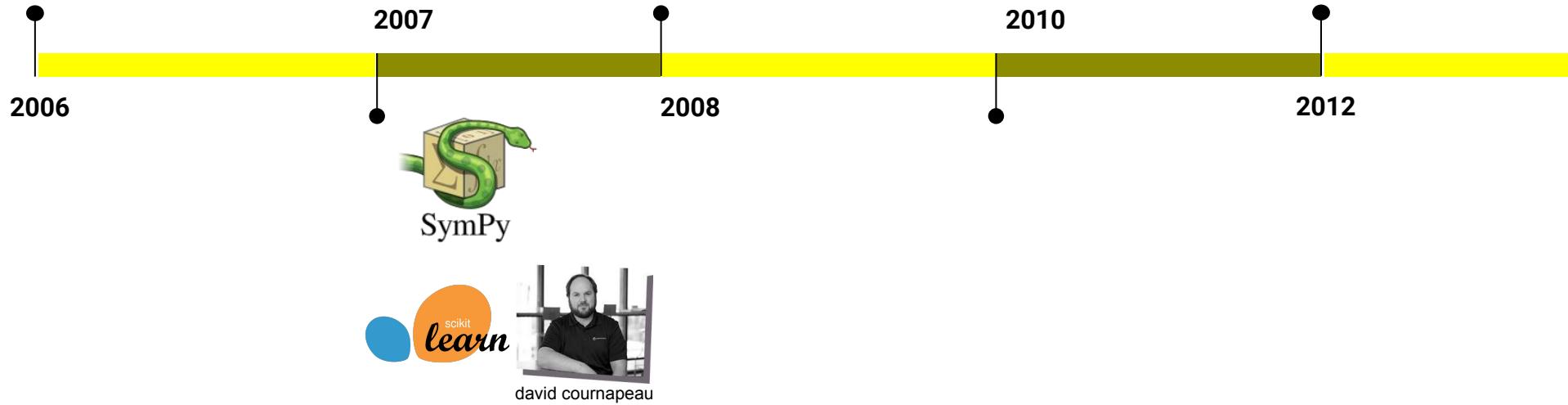
# Història SciPy



Travis Oliphant



Wes McKinney

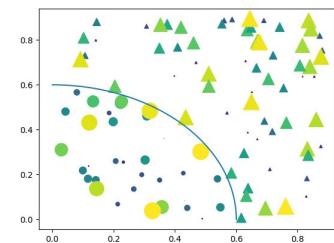
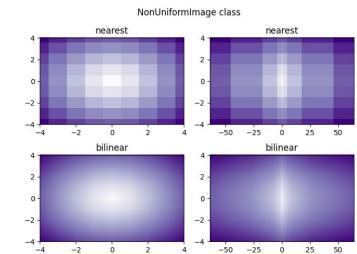
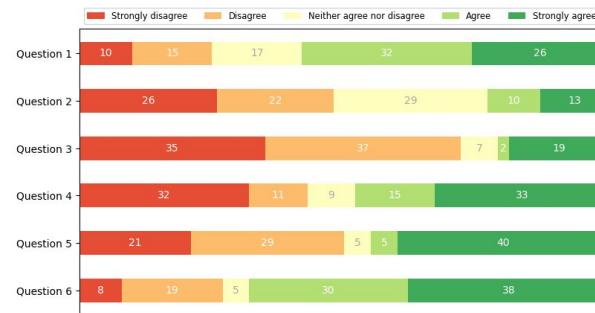
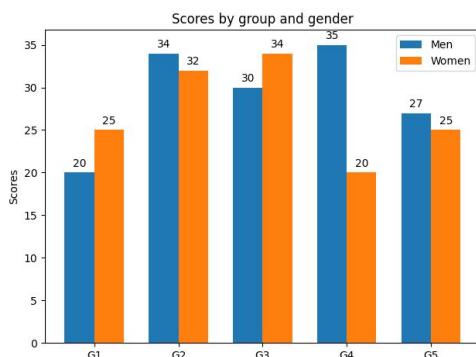


# Què és matplotlib?

Matplotlib és una biblioteca de programari per a generar gràfiques.

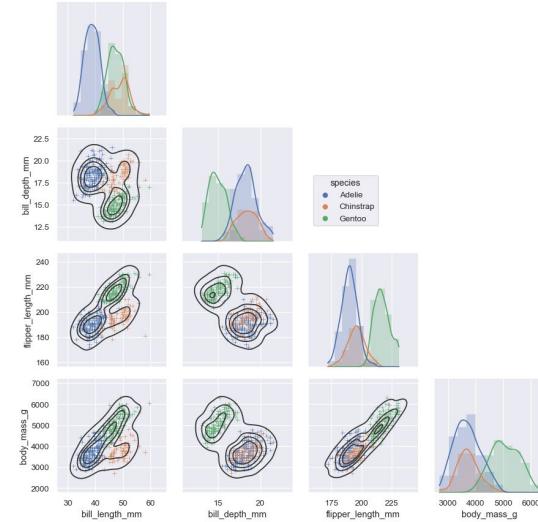
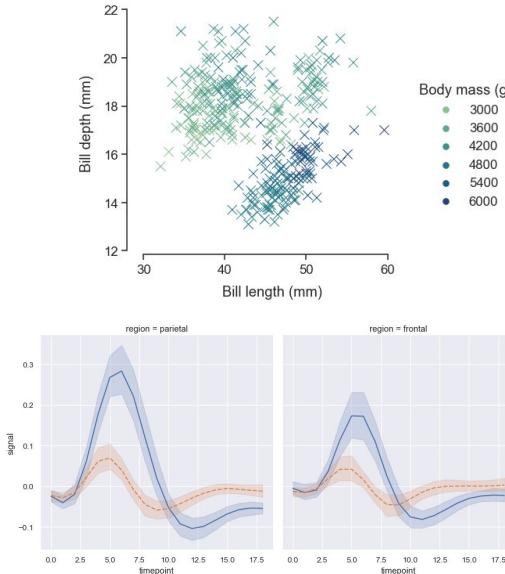
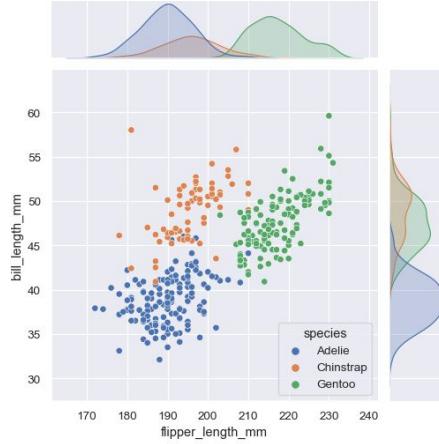
Pyplot es una forma interactiva de utilitzar Matplotlib.

És el més utilitzat



# Què és seaborn?

Seaborn es una altra llibreria per fer gràfics estadístics en Python. No està dins scipy, pero per sota fa servir matplotlib i s'integra molt fàcilment amb estructures de pandas. L'aparença del gràfics és molt més moderna.



# Què és numpy?

Numpy és una extensió de Python, que li agrega major suport per vectors i matrius, constituint una biblioteca de funcions matemàtiques d'alt nivell per operar amb aquests vectors o matrius.

- El nucli és l'objecte *ndarray*, una **matriu n-dimensional** d'un mateix tipus de dada
- Com que Python està implementat com un intèrpret i no com un compilador, els algorismes matemàtics escrits en Python generalment s'executen més lentament
- NumPy vol solucionar aquest problema per a algorismes numèrics creant operadors i funcions eficients per a vectors multidimensionals. Això fa que qualsevol algorisme que es pugui expressar primàriament com a operacions amb vectors i matrius **pot executar-se tant ràpid** com ho faria l'algorisme equivalent en llenguatge C.

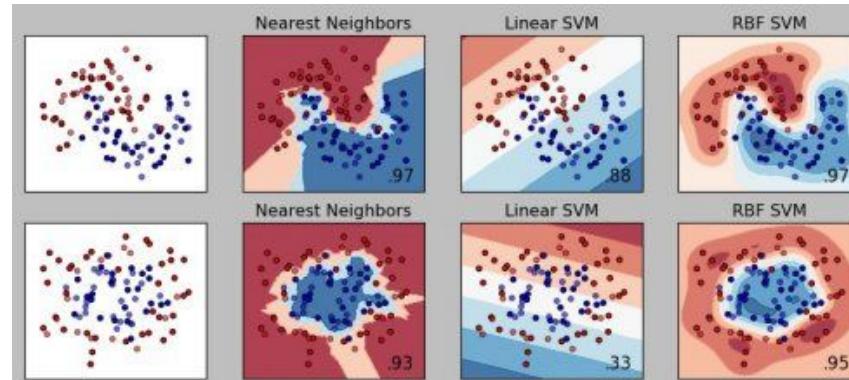
```
>>> a[0,3:5]
array([3,4])
>>> a[4:, 4:]
array([28, 29,
 34, 35])
>>> a[:, 2]
array([2, 8, 14, 20, 26, 32])
>>> a[2 :: 2, :: 2]
array([12, 14, 16,
 24, 26, 28],
 [30, 32, 34])
```

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31	32	33	34	35

# Què és scikit-learn?

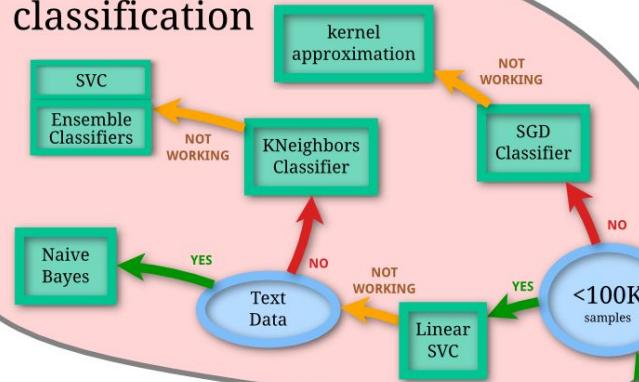
Scikit-learn (també conegut per sklearn) és una extensió del llenguatge Python en forma de biblioteca informàtica que agrega suport en l'àmbit de l'Aprendentatge automàtic.

- Implementa varis algoritmes de **classificació**, **regressió**, **clustering**, **reducció de dimensionalitat**, com SVM, Random Forests, Gradient Boosting, k-Means, PCA..
- Alguns algorismes bàsics s'escriuen a **Cython** per millorar el rendiment. Per exemple, SVM són un wrapper al voltant de LIBSVM; Regressions logístiques al voltant de LIBLINEAR.

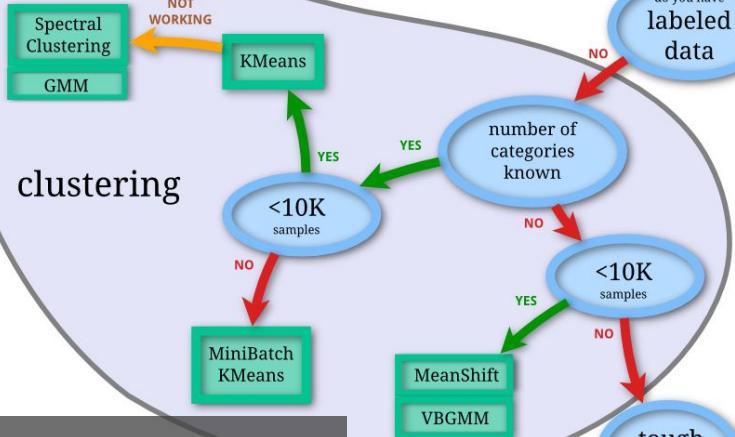


# scikit-learn algorithm cheat-sheet

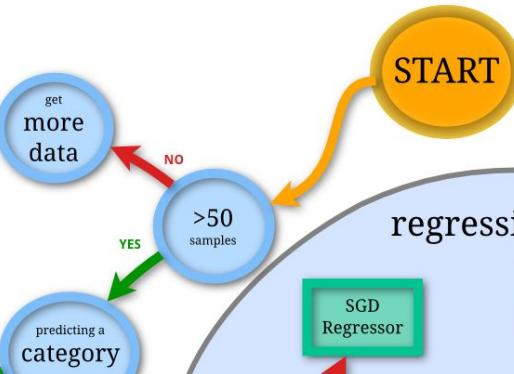
## classification



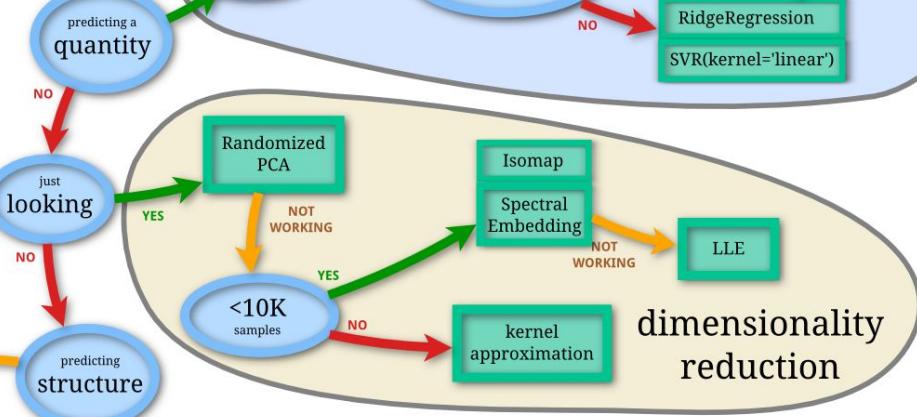
## clustering



Scikit-learn



## regression



## dimensionality reduction

# Què és pandas?

Pandas és una biblioteca de Python per a la manipulació i anàlisi de dades. Està molt ben integrat amb numpy, scikit-learn, matplotlib.. Ofereix **estructures de dades** i operacions per manipular **taules numèriques i sèries temporals**.

El nom deriva del terme "dades del panell", un terme d'econometria per a conjunts de dades que inclouen observacions durant múltiples períodes de temps per als mateixos individus.

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

The diagram illustrates the structure of a Pandas DataFrame. A blue arrow labeled 'Columns' points to the column headers: Name, Team, Number, Position, and Age. An orange arrow labeled 'Rows' points to the index numbers 0 through 6. The table shows data for Boston Celtics players, with some cells highlighted in pink.

# Introducció Eines per Python